Changing Business and Software

James Rumbaugh, IBM Distinguished Engineer
Business is Changing

- Monolithic → distributed
- Tightly controlled → open
- Stable → constantly changing
Where We Are Heading

Case Study: Procure to Pay Process
Where We Are Heading
Case Study: Procure to Pay Process

Change: Customer Order Entry
Where We Are Heading
Case Study: Procure to Pay Process

Change: Shared Service – Marketing, Billing, Receivables
Where We Are Heading

Case Study: Procure to Pay Process

Change: Supplier Handles Inventory
Where We Are Heading

Case Study: Procure to Pay Process

Customer

Division

Shared Service

Supplier

Outsourced

Change: Shipping by FedEx, DHL or UPS
Where We Are Heading
Case Study: Procure to Pay Process

Change: Collections Outsourced
Where We Are Heading

Case Study: Procure to Pay Process

Change: Process Optimization
Why is Incremental Change Desirable?

- More pragmatic approach to re-engineering efforts
  - Gradual improvement
  - Tied to tangible business benefits
- Each function where it best meets business needs:
  - Within the organization
  - From a business partner
  - Completely outsourced
- Tune ongoing business without widespread disruption
IT Architecture is a Choke Point for Business

- Monolithic systems and applications can’t be reused
- Ad hoc integration is difficult to change/maintain
- Lack of standards limits interoperability
- Rigidity makes small improvements hard to justify
But … Technology Applied Correctly can Pave the Way for successful Business Innovation

- Standards (including open source) for interoperability
- Self-defined, loosely coupled interfaces
- Tools to visualize and integrate existing assets
- Model Driven Architecture (MDA)
- Declarative specifications and languages
- Compositional techniques like Aspect Oriented Programming
Tying Business Models to the Supporting IT Architecture

Flexible Business

Transformation
Business Process Outsourcing
Mergers, Acquisitions & Divestitures

Requires

Flexible IT

On Demand Operating Environment

Services Oriented Architecture (SOA)

Development
Software Development

Infrastructure
Integration

Management
Infrastructure Management

Composable Processes (CBM)
Component Business Modeling

Composable Services (SOA)
Service Oriented Architecture
Three Key Concepts

- Component Business Modeling (CBM)
- Service Oriented Architecture (SOA)
- Model Driven Architecture (MDA)
Component Business Models Permit Change

- Identify business transformation opportunities at the business process level
- Make new business process composable and flexible
- Model the business as components with defined services that the business provides or consumes
- Services are provided by IT systems, sourced from providers or supported manually
Business Components

- Business performs *activities*
  - Ex: Rent vehicle
- Activities use *resources*
  - Ex: Rental car, parking spot, agent
- A *business component* is a set of activities and resources
  - Ex: Rental location owns cars and rents and checks in cars
  - It is a virtual business unit that can operate independently
Dependencies among Components

- Activities depend on other components
  - Resource usage
  - Control and information flows
- Goal: restrict dependencies among components
  - Regroup activities and resources
  - Channel them through services
Services

- A service is an explicit interface with no “back doors”
- Service has a defined contract
  - Inputs and outputs
  - Interaction protocol
  - Performance
- Components linked by services are substitutable
A Business Component offers services to other components

Component Name
Market Segment Planning
Description
To analyze markets and derive targets

Required Services
- Business Plans
- Market Events

Provided Services
- Product Portfolio Updates
- Tracking Models & Targets
Business components use each other’s services.

Component Name: Market Segment Planning
Description: To analyze segments and derive targets

Component Name: Business Strategy
Description: Define business strategy

Component Name: Segment Tracking
Description: Track target segments

From "Product Management" business component

- Business Plans
- Market Events
- Product Portfolio Updates
- Product Portfolio Updates
- Tracking Models & Targets
A Business Component Map is a tabular view of the business components in scope.

<table>
<thead>
<tr>
<th>Business Administration</th>
<th>New Business Development</th>
<th>Relationship Management</th>
<th>Servicing and Sales</th>
<th>Product Fulfillment</th>
<th>Financial Control and Accounting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directing</td>
<td>Business Planning</td>
<td>Sector Planning</td>
<td>Account Planning</td>
<td>Sales Planning</td>
<td>Portfolio Planning</td>
</tr>
<tr>
<td>Controlling</td>
<td>Business Unit Tracking</td>
<td>Sector Management</td>
<td>Relationship Management</td>
<td>Sales Management</td>
<td>Fulfillment Planning</td>
</tr>
<tr>
<td></td>
<td>Staff Appraisals</td>
<td>Product Management</td>
<td>Credit Assessment</td>
<td>Sales</td>
<td>Fulfillment Planning</td>
</tr>
<tr>
<td>Executing</td>
<td>Staff Administration</td>
<td>Product Directory</td>
<td>Credit Administration</td>
<td>Sales</td>
<td>Product Fulfillment</td>
</tr>
<tr>
<td></td>
<td>Product Administration</td>
<td>Marketing Campaigns</td>
<td>Contact Routing</td>
<td>Customer Dialog</td>
<td>Document Management</td>
</tr>
</tbody>
</table>
Building an Architecture

Step 1: Break down your business into components
- Decide what is strategically important
- Prioritize and scope your transformation projects

Step 2: Define a Service Model
- Identify services based on business components
- Make decisions based on architectural criteria

Step 3: Implement a Service Architecture
- Develop a service-oriented architecture to support the componentized business

Business-Aligned IT Architecture
Service Oriented Architecture (SOA)

- Flexible connectivity:
  - Represent application or resource as a service with a standardized interface
  - Enable exchange of structured information (messages, documents, ‘business objects’)
  - Mediate message exchange through a connectivity layer (enterprise service bus)
  - Provide on-ramps to the bus for legacy application environments

- Combines new and existing applications to address changing business needs
- Facilitates the management of business performance, quality of service, and dynamic monitoring
Service Component Conceptual Model

Module, Import, and Export

MyValueModule

Service Component MyValue

Service Import StockQuote

Service Export MyValue

Service Import CustomerInfo

Service Export Customer Info

CustomerInfoModule

ESB

Web

(C) Copyright IBM Corporation 2005
SOA Defining Concepts

Universal Connectivity: Integrates most diverse environments, bridging protocols, languages, platforms, APIs and messaging paradigms – providing scale and scope of integration required by today’s extended enterprise

Service Orientation: Facilitates loose coupling between software components

Incremental Integration: Start small and plug in capability to enrich your implementation as needs dictate

Event Orientation: Decouples applications that publish business events from subscribing applications

Flexibility: A variety of options for persistence, reliability, security, availability...

Open, standards-based: Open APIs and protocols support the interoperability and substitution of middleware from multiple vendors
A SOA has Multiple Layers

- Business Process
- Process Choreography
- Services
- Simple and Composite Services
- Components
- Enterprise Components
- Existing Resources
  - Custom Application
  - Package
- QoS, Security, Management & Monitoring (Infrastructure Service)
- Integration Architecture

Service Consumer

Service Provider

Service Modeling
Model Driven Architecture (MDA)

- Build high-level models
  - Expressed in domain concepts
- Generate platform-specific code
  - Into specified architectures
- Using automated tools
  - Based on standards
MDA Approach

- Domain model
  - Captures business knowledge
- Implementation profile
  - Platform
  - Technology decisions
- Design model
  - Captures implementation decisions
  - Translator incorporates platform knowledge
MDA Requires

- Domain models
  - UML or domain-specific language (DSL)
- Architecture frameworks
  - Make assumptions about infrastructure
  - Use them to remove details from models
- Automated tools
  - Modeling tools
  - Translators
Domain Models

- Syntax and semantics for a particular purpose
  - Capture business-level content
  - More intuitive syntax
- Reduce mindless repetition
  - Predefined control and data patterns
  - Eliminate repetitious control specification
- Two approaches:
  - UML Profile
  - Domain specific language (DSL)
Architecture

- Decomposition into subsystems
- Topology
- Interaction rules
- Data formats
- Resource management
- Hooks for future extensions
- Scaffolding for testing
Architecture Framework

- Framework = reusable architecture pattern
- Includes:
  - Skeleton execution environment
  - Predefined subsystems and topology
  - Interaction and data rules, formats, interfaces
  - Attachment points for plug-ins
  - Component libraries of useful functionality
  - Sample applications
- Integrated development environments (IDE)
  - Eclipse, .NET, J2EE
Framework Advantages

- Enable domain specific languages (DSL) to optimize repeated patterns
- Ensure consistency
- Excellent for incremental design
- Enforce design trade-offs
- Supports static and dynamic analysis
Automated Tools

- Modeling tools
  - Organize large models
  - Check correctness
  - Enable teamwork

- Translators
  - Apply patterns
  - Generate repetitious code
  - Key to MDA
Iterative Development

- start
- Iterative process
- iteration
- target

waterfall process

failure
Iteration

- Common baseline
  - No divergent versions or schedules

- Specific goals
  - Set priorities by adjusting goals

- Specific deadline
  - All iterations about the same length

- At the end of an iteration:
  - Check in everything
  - Build an executable system
  - Test and examine it
  - Reevaluate priorities
Iterative Development Process

- Series of iterations allows mid-course corrections
- Executable releases permit constant evaluation
- Attacking problems early reduces risk
Summary

- Business is becoming distributed
  - Component Business Model
- IT architecture must support business models
  - Service Oriented Architecture
- Automate business-to-architecture mapping
  - Model Driven Architecture
- Avoid monolithic development
  - Iterative development process